Étude de réseaux de Thomas par validation de propriétés LTL pour Pseudomonas aeruginosa

Emmanuelle Gallet, Matthieu Manceny, Pascale Le Gall, Paolo Ballarini

Étude de réseaux de Thomas par validation de propriétés LTL pour *Pseudomonas aeruginosa*

Emmanuelle Gallet [1], **Matthieu Manceny** [2], **Pascale Le Gall** [1], **Paolo Ballarini** [1]

1. *Laboratoire MICS, CentraleSupélec, Grande Voie des Vignes 92295 Châtenay-Malabry* CEDEX, *France*
{emmanuelle.gallet,pascale.legall,paolo.ballarini}@centralesupelec.fr

2. *Laboratoire LISITE, ISEP, 28 Rue Notre-Dame-des-Champs 75006 Paris, France*
matthieu.manceny@isep.fr

RÉSUMÉ. *L'évolution des concentrations de protéines caractérise le comportement d'un réseau de régulation génétique (GRN). Elle dépend de la connaissance de paramètres biologiques qui sont en pratique nombreux et difficiles d'accès. Nous proposons une méthode d'inférence de paramètres biologiques de modèles discrets des GRN, s'appuyant sur l'adaptation des algorithmes de model checking LTL. Nous introduisons une structure paramétrée générique (PGRN). En exprimant les connaissances biologiques sous forme de formules LTL, les valeurs des paramètres biologiques sont calculées à l'aide de contraintes caractérisant les cycles acceptants du produit entre un automate de Büchi et un PGRN. Nous illustrons notre méthode avec une étude de cas portant sur l'inductibilité de la cytotoxicité de la bactérie P. aeruginosa.*

ABSTRACT. *The evolution of concentration levels of proteins characterizes the behavior of a genetic regulatory network (GRN). It depends on the knowledge of biological parameters, which are in practice numerous and difficult to know. We propose a method for inferring biological parameters of discrete models of GRN, by adapting algorithms of LTL model checking. We introduce a generic parameterized structure (PGRN). By expressing the biological knowledge in the form of LTL formulas, values of biological parameters become solutions of the constraints characterizing accepting cycles for the product of a Büchi automaton and a PGRN. We illustrate our method with a case study on the inducibility of the cytotoxicity of bacteria P. aeruginosa.*

MOTS-CLÉS : *model-checking LTL, exécution symbolique, résolution de contraintes, modèles discrets de réseau de régulation génétique, Pseudomonas aeruginosa.*

KEYWORDS: *LTL model checking, symbolic execution, constraint solving, discrete models of genetic regulatory network, Pseudomonas aeruginosa.*

1. Introduction

Dans cet article, nous nous intéressons aux réseaux de régulation génétique (GRN pour *Genetic Regulatory Networks*) qui coordonnent les interactions biologiques entre protéines et gènes. L'expression des gènes est régulée par la présence ou l'absence de protéines régulatrices. L'interdépendance des régulations activatrices et inhibitrices rend ces systèmes intrinsèquement complexes. Différentes approches – mathématiques et informatiques – de modélisation ont été proposées pour aider à une meilleure compréhension du fonctionnement des GRN, à savoir, à une meilleure connaissance des lois qui régissent l'évolution temporelle des concentrations de protéines dans la cellule, exprimées à partir des gènes du GRN. La mise en œuvre de ces méthodes nécessite la prise en compte de paramètres biologiques qui, en pratique, sont très nombreux et difficiles à estimer. Deux tendances se dessinent : d'une part, fixer l'ensemble des paramètres, soit *a priori* soit en faisant appel à des connaissances d'experts, et proposer des simulations fines dont l'analyse permet de conforter ou réviser les hypothèses initiales quant aux valeurs des paramètres choisis ; d'autre part, proposer des méthodes dont l'objectif est précisément d'aider les biologistes à inférer les valeurs des paramètres en question pour leur donner l'opportunité d'extraire des prédictions biologiques intéressantes à partir des modèles issus de l'instanciation des paramètres. Dans cet article, nous nous inscrivons dans cette deuxième voie.

Les premières modélisations discrètes des GRN ont été booléennes (Kauffman, 1969) : l'action d'une protéine sur un gène, *i.e.* régulation positive ou négative, est directement corrélée à sa présence ou son absence dans la cellule. L'expression d'un gène résulte alors de la combinaison des actions des protéines régulatrices, et dépend de l'issue de la compétition entre ses protéines activatrices et inhibitrices. La modélisation discrète introduite par René Thomas (Thomas, d'Ari, 1990 ; Thieffry *et al*., 1993 ; Thieffry, Thomas, 1995) est multivaluée. L'effet régulateur d'une protéine sur un gène devient effectif lorsque son niveau de concentration atteint un certain seuil, caractéristique du gène cible. Ainsi, les concentrations des protéines dans la cellule peuvent être discrétisées en autant de niveaux qu'il y a d'effets régulateurs possibles pour la protéine. Les modélisations discrètes assimilent un gène à la protéine exprimée à partir de ce gène, ainsi les évolutions temporelles des (niveaux discrets des) concentrations de protéines, aussi connues sous le nom de *dynamiques*, caractérisent le fonctionnement des GRN. Ces dynamiques sont assujetties à la connaissance de paramètres biologiques indicatifs des effets de compétition entre protéines activatrices et inhibitrices d'un même gène. En l'absence de telles connaissances biologiques, on ignore le bilan d'une situation de compétition, l'étude des dynamiques nécessite d'envisager toutes les configurations possibles. Le nombre de combinaisons croît de façon exponentielle en fonction du nombre de protéines régulatrices d'un même gène.

La modélisation discrète des GRN proposée par R. Thomas présente l'avantage de se prêter à des analyses de nature qualitative par l'application de méthodes formelles, en particulier du *model checking* (Baier, Katoen, 2008). En effet, lorsque les connaissances biologiques à propos d'un GRN peuvent être formulées sous la forme d'une propriété relevant d'une logique temporelle (e.g. LTL, CTL), une étape de

model checking entre une dynamique particulière (issue d'un choix de valeurs pour les paramètres) et la formule temporelle permet d'affirmer si le jeu de valeurs associé aux paramètres est admissible au vu des connaissances biologiques. De nombreux travaux ont été entrepris afin de spécialiser les techniques de model checking en lien avec une particularité des GRN, à savoir que la taille des modèles (*i.e.* des dynamiques) reste raisonnable, alors que le nombre des dynamiques (*i.e.* nombre de paramètres et choix possible de valeurs pour ces paramètres) devient vite ingérable. Dans (Bernot *et al*., 2004), les auteurs ont mis en place une approche systématique consistant à construire toutes les dynamiques possibles pour un GRN donné et à confronter, l'une après l'autre, chacune des dynamiques à des connaissances biologiques exprimées en logique CTL. L'approche a été réalisée au sein de l'outil SMBioNet (Richard, 2010) et a été illustrée dans (Bernot *et al*., 2004 ; Filopon *et al*., 2006). Ces derniers travaux mettent en avant le double enjeu de chercher les valeurs des paramètres biologiques une fois les régulations équipées d'une valeur de seuil, mais aussi de considérer les différentes possibilités d'associer une valeur de seuil à une régulation. Force est de constater que même si les biologistes connaissent la nature d'une régulation (activation ou inhibition), la valeur du seuil associé donne souvent lieu à discussion. En effet, lorsqu'une protéine agit sur deux gènes, le seuil indique à partir de quel niveau de concentration la régulation a lieu, et donc quelle interaction a la précédence, ce qui est souvent discutable. Il devient nécessaire d'envisager plusieurs configurations de valeurs de seuils dans les GRN. Afin de réduire le nombre de dynamiques, l'approche proposée dans (Khalis *et al*., 2009) enrichit la modélisation en intégrant les complexes protéiques issus des interactions protéines-protéines, introduisant des égalités entre paramètres et réduisant ainsi le choix de leurs valeurs possibles. Dans (Klarner *et al*., 2012 ; Barnat *et al*., 2012), l'ensemble des paramétrisations possibles est encodé par un vecteur de booléens, un par dynamique. En exploitant les versions efficaces des opérations booléennes bit à bit et à l'aide de techniques de parallélisation, l'algorithme de model checking LTL est optimisé pour le cas particulier de *séries temporelles*. Dans (Corblin *et al*., 2009 ; 2010), des techniques de programmation logique avec contraintes (PLC) sont utilisées en manipulant les paramètres biologiques comme des variables logiques définies sur des domaines finis. Cependant, les seuls chemins considérés sont des chemins finis, et par conséquent, le pouvoir d'expression des logiques temporelles n'est pas accessible.

Comme (Klarner *et al*., 2012), notre approche est une extension du model checking LTL et comme les approches à base de PLC, nous manipulons les paramètres biologiques sous forme de variables logiques d'un système de contraintes. Cet article est une extension des travaux (Mateus *et al*., 2007 ; Gallet *et al*., 2014a ; 2014b), La contribution est double. Premièrement, nous décrivons les différentes optimisations mises en jeu dans l'outil SPuTNIk pour maîtriser l'explosion combinatoire des structures de données et les temps de calcul. Deuxièmement, nous présentons une étude de cas portant sur l'inductibilité de la cytotoxicité de la bactérie *Pseudomonas aeruginosa*. Nous reprenons une étude faite par (Filopon *et al*., 2006) avec l'outil SMBioNet (Bernot *et al*., 2004) sur un GRN réduit composé uniquement de trois gènes, que nous étendons ensuite en un GRN composé de sept gènes.

Le papier est organisé selon le plan suivant. Dans la section 2, nous reformulons la description logique du modèle de Thomas (comme détaillée dans (Barnat *et al.*, 2012 ; Bernot *et al.*, 2004 ; Corblin *et al.*, 2009 ; Gallet *et al.*, 2014b ; 2014a)) en encodant l'ensemble des dynamiques sous la forme d'une structure paramétrée par les paramètres biologiques, appelée GRN Paramétré. La section 3 introduit notre adaptation du model checking LTL et détaille les mécanismes d'exécution symbolique et de résolution de contraintes mis en jeu pour détecter les cycles paramétriques acceptants. L'étude de cas est détaillée dans la section 4.

2. Modélisation paramétrique d'un GRN

2.1. Graphe d'interaction

Comme indiqué en introduction, un ensemble de gènes/protéines régulés de manière interdépendante est appelé *réseau de régulation génétique* (GRN). L'expression d'un gène dépend de la présence de protéines régulatrices : si la concentration de ces protéines est suffisante, elles peuvent activer ou inhiber la transcription du gène et donc la synthèse de la protéine associée au gène ; nous identifierons une protéine au gène à partir duquel elle est synthétisée. La concentration des protéines dans la cellule (qui est une donnée continue) est discrétisée en un nombre fini de *niveaux d'expression* qualitatifs : chaque valeur de niveau correspond à un seuil de concentration minimale à atteindre pour réguler un gène cible et ainsi, modifier le niveau d'expression de ce dernier (le niveau 0 signifiant que la concentration de la protéine est trop faible pour que celle-ci régule un gène). Le nombre (et donc la valeur maximale) de niveaux d'expression d'un gène donné est inférieur ou égal au nombre de gènes qu'il régule. Un GRN est classiquement représenté par un *graphe d'interaction*.

DÉFINITION 1 (Graphe d'interaction). — *Un* graphe d'interaction *est un graphe orienté étiqueté $G = (V, E, S, T)$ où V est un ensemble fini de nœuds, assimilés à des* gènes *et $E \subseteq V \times V$ est l'ensemble des arcs du graphe, représentant les* interactions *entre les gènes.*
$S : E \to \{+, -\}$ et $T : E \to \mathbb{N}^+$ sont deux applications, associant respectivement à chaque interaction $(i, j) \in E$ l'effet de i sur j ("+" pour activation *et "−" pour* inhibition*) et le* seuil de régulation*, autrement dit le niveau minimum que doit atteindre i pour réguler j. De plus, les seuils de régulation respectent la propriété suivante :*
$\forall (i, j) \in E, T(i, j) > 1 \Rightarrow (\exists (i, j') \in E, T(i, j') = T(i, j) - 1)$.

La propriété portant sur les seuils de régulation indique que pour un gène i ayant un niveau d'expression maximal donné, tous les niveaux intermédiaires sont utilisés pour au moins une autre interaction dont i est la source.

EXEMPLE 2. — Étude de cas : inductibilité de la cytotoxicité de *P. aeruginosa*

Nous considérons une étude de cas portant sur l'inductibilité de la cytotoxicité de *Pseudomonas aeruginosa*. Cette bactérie possède un système de production et de sécrétion de toxines (appelé *Système de Sécrétion de Type III* ou SST3) qui, une fois activé, peut provoquer la mort ou le dysfonctionnement de cellules cibles.

Des expériences biologiques ont montré l'existence de deux types de souches différentes de la bactérie. Dans le cas de souches dites *inductibles*, l'activation de SST3 conduit à un état *induit*, dans lequel les toxines sont produites et injectées dans la cellule cible. *In vivo*, cette activation est déclenchée par le contact entre la bactérie et la cellule cible ; *in vitro*, l'activation est causée par l'appauvrissement du milieu de culture en calcium. Par contre, dans le cas de souches dites *non inductibles*, le SST3 n'est jamais activé, en dépit des signaux *in vivo* ou *in vitro*. Comme aucune mutation entre ces souches n'a été trouvée, une des hypothèses permettant d'expliquer ce phénomène repose sur la modification du phénotype cytotoxique/non cytotoxique (résultant de l'expression des gènes) en fonction de l'environnement, via un mécanisme appelé *«switch» épigénétique*. Les switches épigénétiques s'observent en présence d'au moins un circuit de rétroaction positif, c'est-à-dire d'un cycle $i_1 \to i_2 \to \ldots \to i_k = i_1$ comportant un nombre pair d'inhibitions.

Le GRN pilotant l'inductibilité de la cytotoxicité de *P. aeruginosa* peut être réduit à trois gènes, comme présenté dans l'article (Filopon *et al.*, 2006). Ce GRN est composé des gènes [1] suivants : ExsA, ExsD et le complexe ToxSST3 (qui représente un complexe de gènes composé des gènes codant les exotoxines sécrétées par SST3 et ceux codant l'appareil du SST3 lui-même). ExsA est un activateur de l'ensemble des toxines du complexe ToxSST3 ; nous savons également que sa synthèse est activée par lui-même mais est inhibée par ExsD. Dans ce modèle simplifié, nous considérons que le milieu est pauvre en calcium, ce signal d'activation de SST3 étant nécessaire pour avoir un état induit. Plus exactement, nous considérons que le signal vient d'être ajouté au temps initial de l'étude, nous plaçant dans un système exempt de toxines.

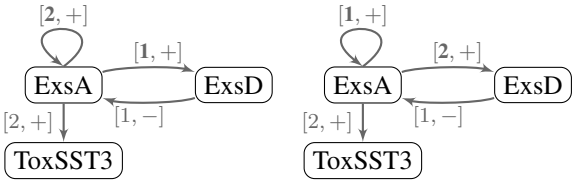
Avec ces informations, nous pouvons établir un premier graphe d'interaction de ce GRN, G_{1a}, présenté en figure 1a. ExsA active ExsD dès qu'il atteint son premier niveau d'expression ; il active également sa propre expression ainsi que l'expression de ToxSST3 à partir de son niveau 2. Cependant, les biologistes ignorent quel seuil de régulation entre celui de ExsA $\to$ ExsA et celui de ExsA $\to$ ExsD est le plus haut [2]. Ainsi, tout comme (Filopon *et al.*, 2006), nous considérons aussi la deuxième version du graphe d'interaction, G_{1b}, présentée en figure 1b.

□

2.2. Dynamiques et paramètres biologiques

Par la suite, pour un GRN $G = (V, E, S, T)$, le niveau d'expression d'un gène i sera noté x_i. La connaissance de l'ensemble des niveaux d'expression des gènes à un instant donné définit un *état* noté $x = (x_i)_{i \in V}$. L'évolution de x au cours du temps est appelée *dynamique* du GRN G. Nous nous plaçons dans le cadre de dynamiques asynchrones et pas-à-pas : à partir de chaque état, le prochain état est identique à l'état courant à une unité de valeur d'un gène (unique) près.

1. Les protéines synthétisées à partir de ces gènes seront nommées de la même manière.
2. Nous savons par contre que ExsA doit être à un niveau de concentration élevé pour activer les toxines.



(a) Version 1 : G_{1a} (b) Version 2 : G_{1b}

Figure 1. Les deux versions minimales du graphe d'interaction de P. aeruginosa

Chaque x_i évolue en fonction des niveaux d'expressions des gènes régulateurs de i notés $V^-(i) = \{j | j \in V, (j,i) \in E\}$. Plus précisément, pour un état donné $x = (x_i)_{i \in V}$, l'évolution de x_i dépend uniquement des gènes j issus de $V^-(i)$ dont le niveau d'expression x_j est supérieur ou égal au seuil $T(j,i)$, *i.e.* les gènes de $\{j \mid j \in V^-(i), x_j \geq T(j,i)\}$. Dans ce cas, les gènes j sont dits *effectifs* pour i. Pour tout sous-ensemble ω des régulateurs de i ($\omega \in 2^{V^-(i)}$), il existe une constante $K_i(\omega)$ dans $\mathbb{N}$ qui représente le niveau d'expression vers lequel tend x_i dans un état donné $x = (x_i)_{i \in V}$, tel que $\forall j \in \omega, x_j \geq T(j,i)$ et $\forall j' \in V^-(i) \backslash \omega, x_{j'} < T(j',i)$. Si $x_i < K_i(\omega)$ alors x_i peut augmenter, si $x_i > K_i(\omega)$ alors x_i peut diminuer, enfin si $x_i = K_i(\omega)$ alors x_i reste constant. Ainsi, une dynamique d'un GRN peut être décrite par *un jeu de paramètres* $K_i(\omega)$, spécifiant l'évolution de chaque niveau d'expression en fonction des gènes effectifs.

DÉFINITION 3 (Paramètres biologiques). — *Soit $G = (V, E, S, T)$ un graphe d'interaction.*
Les paramètres biologiques *de G sont les éléments de* $\{K_i(\omega) \mid i \in V, \omega \subseteq V^-(i)\}$.

EXEMPLE 4 (Paramètres biologiques). — Les paramètres biologiques de G_{1a} et G_{1b} (figure 1) sont : $K_{\text{ExsA}}(\{\})$, $K_{\text{ExsA}}(\{\text{ExsA}\})$, $K_{\text{ExsA}}(\{\text{ExsD}\})$, $K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\})$, $K_{\text{ExsD}}(\{\})$, $K_{\text{ExsD}}(\{\text{ExsA}\})$, $K_{\text{ToxSST3}}(\{\})$ et $K_{\text{ToxSST3}}(\{\text{ExsA}\})$. Considérons l'état $(x_{\text{ExsA}}, x_{\text{ExsD}}, x_{\text{ToxSST3}}) = (1, 0, 0)$. L'ensemble des prédécesseurs effectifs de ExsA est soit vide pour la version G_{1a} (et tend donc vers $K_{\text{ExsA}}(\{\})$), soit égal à $\{\text{ExsD}\}$ pour la version G_{1b} (et tend donc vers $K_{\text{ExsA}}(\{\text{ExsD}\})$). Quelle que soit la version considérée, ToxSST3 n'a pas de prédécesseur effectif et tend donc vers $K_{\text{ToxSST3}}(\{\})$, et ExsD possède un prédécesseur effectif (ExsA) et tend vers $K_{\text{ExsD}}(\{\text{ExsA}\})$. □

Par construction, les paramètres $K_i(\omega)$ dénotent des niveaux de concentration et appartiennent donc au domaine licite pour le gène i, c'est-à-dire à l'un des niveaux discrets de i, compris entre 0 et $max_{(i,j) \in E} T(i,j)$. Ainsi, le nombre possible de jeux de paramètres biologiques dépend du nombre de gènes composant le GRN et du nombre de régulateurs de chacun de ses gènes. Pour un graphe d'interaction $G = (V, E, S, T)$, il est de l'ordre de $\prod_{i \in V} (|V^+(i)| + 1)^{2^{|V^-(i)|}}$, avec $V^+(i) = \{j | j \in V, (i,j) \in E\}$ l'ensemble des gènes régulés par i.

EXEMPLE 5 (Nombre de jeux de paramètres biologiques). — Pour G_{1a} et G_{1b}, le nombre de jeux de valeurs possibles pour les paramètres $K_i(\omega)$ est égal à 1296. □

2.3. GRN paramétré

En pratique, les paramètres biologiques dépendent de considérations biologiques et sont pour la plupart inconnus. Afin de pouvoir considérer conjointement toutes les dynamiques possibles, nous introduisons un méta-modèle, appelé *GRN paramétré* (ou Parametric GRN, PGRN). Un PGRN contient deux familles de symboles : la famille des paramètres biologiques $K_i(\omega)$ qui représentent des constantes biologiques inconnues, et la famille des variables d'état x_i qui représentent les niveaux d'expression des gènes i dans V, et qui peuvent évoluer, c'est-à-dire augmenter ou diminuer d'une unité à la fois. Pour chaque gène i, un PGRN contient deux transitions, une pour l'augmentation et une pour la diminution de x_i.

Pour $\omega \in 2^{V^-(i)}$, notons $P_i(\omega)$ la proposition logique correspondant à l'ensemble des états $x = (x_i)_{i \in V}$ dans lesquels ω est l'ensemble des régulateurs effectifs du gène i : $P_i(\omega) \equiv (\bigwedge_{j \in \omega} x_j \geq T(j,i)) \wedge (\bigwedge_{j \in V^-(i) \setminus \omega} x_j < T(j,i))$. x_i ne peut donc augmenter que si la *garde* $Increase(i) = \vee_{\omega \subset V^-(i)} Increase(i,\omega)$ est vraie, avec $Increase(i,\omega) = (P_i(\omega) \wedge x_i < K_i(\omega))$. De même, la diminution de x_i est conditionnée par la garde $Decrease(i) = \vee_{\omega \subset V^-(i)} Decrease(i,\omega)$, où $Decrease(i,\omega) = (P_i(\omega) \wedge x_i > K_i(\omega))$. Enfin, le niveau d'expression du gène i reste constant pour la condition $Stable(i) = \vee_{\omega \subset V^-(i)} (P_i(\omega) \wedge x_i = K_i(\omega))$.

DÉFINITION 6 (GRN Paramétré). — *Soit $G = (V, E, S, T)$ un graphe d'interaction. Le GRN paramétré associé à G est un couple (St, Tr) avec $St = \{I, F\}$ un ensemble de deux sommets (I le sommet transitoire et F le sommet stable), et Tr un ensemble de transitions. Une transition de Tr est de la forme (s, g, a, s') avec s et s' des sommets dans St, g la garde et a l'affectation et elle respecte l'un des formats suivants :*

- $(I, Increase(i), x_i \uparrow, I)$ *avec* $i \in V$,
- $(I, Decrease(i), x_i \downarrow, I)$ *avec* $i \in V$,
- $(I, \wedge_{i \in V} Stable(i), id, F)$ *avec* $i \in V$ *et où* id *est l'application identité,*
- $(F, \top, id, F)$ *où* $\top$ *représente la garde toujours vraie*

où $x_i \uparrow$ (resp. $x_i \downarrow$) correspond à l'affectation où seule la composante x_i des états change, en augmentant de un, i.e. $x_i \mapsto x_i + 1$ (resp. en diminuant de un, i.e. $x_i \mapsto x_i - 1$).

EXEMPLE 7 (GRN Paramétré). — La figure 2 présente la forme générale du PGRN associé aux graphes d'interaction de la figure 1. Il y six (deux fois le nombre de gènes) transitions de I à I, une transition de I à F, et une transition de F à F. Les gardes des transitions dépendent de la version du graphe d'interaction. Par exemple, les conditions d'augmentation de ExsD dépendent du seuil de ExsA $\rightarrow$ ExsD. Ainsi, on obtient pour G_{1a} : $Increase(\text{ExsD}) \equiv (x_{\text{ExsA}} < 1 \wedge x_{\text{ExsD}} < K_{\text{ExsD}}(\{\})) \vee (x_{\text{ExsA}} \geq 1 \wedge x_{\text{ExsD}} < K_{\text{ExsD}}(\{\text{ExsA}\}))$. □

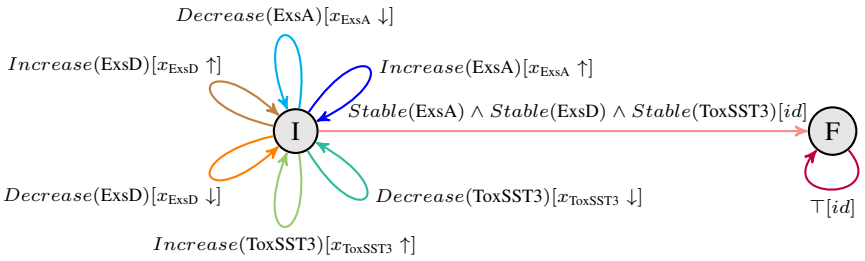


Figure 2. Forme générale du PGRN associé aux graphes d'interaction G_{1a} et G_{1b}

EXEMPLE 8 (Dynamique et graphe d'états). — Considérons le graphe d'interaction G_{1a}. Si l'état $(x_{\text{ExsA}}, x_{\text{ExsD}}, x_{\text{ToxSST3}}) = (1, 0, 0)$ est tel que les conditions $Increase(\text{ExsA})$ et $Increase(\text{ExsD})$ sont satisfaites, alors le modèle peut évoluer soit vers $(2, 0, 0)$ soit vers $(1, 1, 0)$.
La figure 3 présente sous la forme d'un *graphe d'états* la dynamique correspondant au jeu de paramètres suivant : $K_{\text{ExsA}}(\{\}) = 2$, $K_{\text{ExsA}}(\{\text{ExsA}\}) = 2$, $K_{\text{ExsA}}(\{\text{ExsD}\}) = 0$, $K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\}) = 1$, $K_{\text{ExsD}}(\{\}) = 0$, $K_{\text{ExsD}}(\{\text{ExsA}\}) = 1$, $K_{\text{ToxSST3}}(\{\}) = 0$ et $K_{\text{ToxSST3}}(\{\text{ExsA}\}) = 1$. Dans ce graphe orienté, une transition $(x_{\text{ExsA}}, x_{\text{ExsD}}, x_{\text{ToxSST3}}) \rightarrow (x'_{\text{ExsA}}, x'_{\text{ExsD}}, x'_{\text{ToxSST3}})$ existe seulement s'il existe une transition (s, g, a, s') dans le PGRN de G_{1a} telle que $(x_{\text{ExsA}}, x_{\text{ExsD}}, x_{\text{ToxSST3}})$ satisfait g et $(x'_{\text{ExsA}}, x'_{\text{ExsD}}, x'_{\text{ToxSST3}}) = a((x_{\text{ExsA}}, x_{\text{ExsD}}, x_{\text{ToxSST3}}))$. □

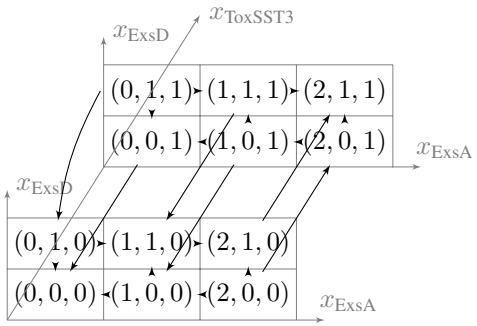


Figure 3. Graphe d'états représentant une dynamique possible de G_{1a}

2.4. Contraintes sur les paramètres

Le GRN paramétré obtenu à partir d'un GRN représente une structure de base que l'on peut compléter avec des informations biologiques. En effet, la plupart des valeurs des paramètres sont inconnues, mais des connaissances biologiques additionnelles

peuvent réduire le nombre de jeux de paramètres admissibles en écartant les impossibilités biologiques. Elles s'expriment sous la forme de contraintes sur les paramètres du PGRN :

DÉFINITION 9 (Contraintes sur les paramètres). — *Soit $G = (V, E, S, T)$ un graphe d'interaction.*

– Contrainte de domaine :

$$\forall i \in V, \forall \omega \subset V^-(i), 0 \leq K_i(\omega) \leq \max_{(i,j)\in E} T(i,j)$$

– Contrainte de définition : $\forall i \in V$, $\forall j \in V^-(i)$, $\forall \omega \subset V^-(i) \setminus \{j\}$,

$$\begin{cases} K_i(\omega) \leq K_i(\omega \cup \{j\}) & \textit{si } S(j,i) = + \\ K_i(\omega) \geq K_i(\omega \cup \{j\}) & \textit{si } S(j,i) = - \end{cases}$$

– Contrainte d'observation : $\forall i \in V$, $\forall j \in V^-(i)$, $\exists \omega \subset V^-(i) \setminus \{j\}$,

$$\begin{cases} K_i(\omega) < K_i(\omega \cup \{j\}) & \textit{si } S(j,i) = + \\ K_i(\omega) > K_i(\omega \cup \{j\}) & \textit{si } S(j,i) = - \end{cases}$$

– Contrainte min/max : $\forall i \in V$

$$\begin{cases} K_i(\{j | j \in V^-(i), S(j,i) = -\}) = 0 \\ K_i(\{j | j \in V^-(i), S(j,i) = +\}) = \max_{(i,j)\in E} T(i,j) \end{cases}$$

Nous avons déjà discuté de la *contrainte de domaine* qui borne les valeurs des paramètres biologiques.
La *contrainte de définition*, ou contrainte de Snoussi (Snoussi, Thomas, 1993), spécifie que si un activateur (respectivement inhibiteur) j d'un gène i devient effectif (*i.e.* son niveau d'expression franchit le seuil de l'interaction $j \rightarrow i$), alors le niveau d'expression de i ne peut pas diminuer (resp. augmenter).
La *contrainte d'observation* indique comment est identifié l'effet d'un régulateur sur sa cible. Si j est un activateur (resp. inhibiteur) de i, alors il existe une situation où l'augmentation du niveau d'expression de j au-dessus de son seuil de régulation conduit à une augmentation (resp. diminution) de x_i.
Enfin, la *contrainte min/max* spécifie que dans un état où tous les activateurs (resp. inhibiteurs) d'un gène sont effectifs et où, simultanément, aucun inhibiteur (resp. activateurs) ne l'est, alors le niveau d'expression du gène régulé tend à être maximum (resp. minimum, c'est-à-dire égal à 0).

EXEMPLE 10 (Contraintes sur les paramètres). — En appliquant les contraintes décrites précédemment aux paramètres de G_{1a}, nous obtenons (après simplification) les conditions suivantes : $K_{\text{ExsA}}(\{\text{ExsA}\}) = 2$, $K_{\text{ExsA}}(\{\text{ExsD}\}) = 0$, $K_{\text{ExsD}}(\{\}) = 0$, $K_{\text{ExsD}}(\{\text{ExsA}\}) = 1$, $K_{\text{ToxSST3}}(\{\}) = 0$, $K_{\text{ToxSST3}}(\{\text{ExsA}\}) = 1$, $(K_{\text{ExsA}}(\{\}) < 2 \vee 0 < K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\}))$ et $(K_{\text{ExsA}}(\{\}) > 0 \vee 2 > K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\})$.
Ainsi, pour G_{1a}, la garde $Increase(\text{ExsD}) \equiv (x_{\text{ExsA}} < 1 \wedge x_{\text{ExsD}} < K_{\text{ExsD}}(\{\})) \vee (x_{\text{ExsA}} \geq 1 \wedge x_{\text{ExsD}} < K_{\text{ExsD}}(\{\text{ExsA}\}))$ du PGRN associé peut être simplifiée en : $(x_{\text{ExsA}} \geq 1 \wedge x_{\text{ExsD}} = 0)$. Au final, seuls 7 jeux de paramètres sont compatibles

avec ces contraintes, sur un total de 1296 jeux correspondants à la seule contrainte de domaine. Ces 7 jeux de paramètres définissent 5 dynamiques différentes [3]. □

Notons que nous n'intégrons pas directement ces contraintes dans la définition du PGRN car elles ne sont pas toujours toutes prises en compte par les biologistes (exception faite de la contrainte sur le domaine). Ainsi, même si ces contraintes sont légitimes, nous pouvons les relâcher sur demande. Dans la suite de l'article, elles seront par défaut utilisées dans le cadre des exemples.

3. Adaptation du model checking LTL au GRN Paramétré

L'approche classique du *model checking LTL* consiste à confronter chaque dynamique avec les comportements observés (*in vivo* ou *in vitro*) ou avec des hypothèses biologiques, exprimés sous la forme de formules temporelles sur les niveaux d'expression des gènes en *Logique Temporelle Linéaire*. A partir d'un automate de Büchi reconnaissant la négation de la formule LTL, un produit entre l'automate de Büchi et une dynamique est construit. On recherche alors s'il existe dans le produit un chemin permettant d'atteindre un cycle contenant au moins un état acceptant. Si un tel chemin existe, c'est la preuve qu'un chemin de la dynamique satisfait la négation de la propriété biologique et par conséquence, la propriété n'est pas satisfaite sur tous les chemins de la dynamique et elle doit donc être exclue des solutions.

Étant donné le grand nombre de dynamiques, cette méthode n'est pas applicable en pratique. Nous proposons de considérer l'ensemble des dynamiques à l'aide du PGRN plutôt que chaque dynamique séparément. Nous construisons ainsi le produit [4] entre le PGRN et l'automate de Büchi considéré. Nous utilisons ensuite des techniques d'exécution symbolique pour rechercher des chemins acceptants. Le résultat de cette recherche prend la forme d'une contrainte telle que seuls les jeux de paramètres vérifiant cette contrainte définissent des dynamiques compatibles avec la propriété biologique.

3.1. Logique LTL, automate de Büchi et Produit

À l'aide de formules LTL, il est possible d'exprimer des propriétés biologiques sur les niveaux d'expression des gènes. Ces formules sont construites à partir d'un ensemble de propositions atomiques sur les niveaux d'expression de la forme $x_i \bowtie c$, avec x_i le niveau d'expression du gène i, $\bowtie \in \{=, \neq, <, >, \leq, \geq\}$ et $c \in \mathbb{N}$. Les formules LTL utilisent également les propositions Vraie ($\top$) et Faux ($\bot$), ainsi que les opérateurs logiques habituels $\{\neg, \wedge, \vee, \Rightarrow\}$ et les opérateurs temporels $\mathbf{X}$ (pour *neXt time*, « à l'instant suivant »), $\mathbf{G}$ (*Globally*, « toujours dans le futur »), $\mathbf{F}$ (*Finally*, « à un moment dans le futur ») et $\mathbf{U}$ (*Until*, « jusqu'à ce que ») (Baier, Katoen, 2008).

3. En effet, les jeux de paramètres tels que $K_{\text{ExsA}}(\{\}) < 2 \wedge K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\}) < 2$ ne définissent que 2 dynamiques différentes (l'une pour $K_{\text{ExsA}}(\{\}) = 0$, l'autre pour $K_{\text{ExsA}}(\{\}) = 1$).

4. Le produit entre un PGRN et un automate de Büchi sera noté « Produit » par la suite.

EXEMPLE 11. — Dans le cas de l'inductibilité de la cytotoxicité de *P. aeruginosa*, deux propriétés biologiques peuvent être exprimées. Dans le cas d'une souche inductible (et dans un milieu pauvre en calcium), la cytotoxicité de la bactérie est toujours activée après un certain temps et ce caractère reste stable ; dans le cas d'une souche non inductible, la bactérie ne peut pas sécréter de toxines. Ces connaissances se traduisent alors en LTL par[5] $\varphi_1 \equiv \mathbf{G}(inductible \Rightarrow \mathbf{FG}\, cytotoxique)$ et $\varphi_2 \equiv \mathbf{G}(\neg inductible \Rightarrow \mathbf{FG}\, \neg cytotoxique)$.

Que recouvrent les propositions $cytotoxique$ et $inductible$? La cytotoxicité correspond à la production de toxines, et donc aux états où $x_{\text{ToxSST3}} = 1$ (autrement dit, $cytotoxique$ coïncide avec l'égalité $x_{\text{ToxSST3}} = 1$). D'après le seuil de l'interaction ExsA $\rightarrow$ ToxSST3 (identique dans les deux versions de graphes d'interaction), un état inductible correspond aux situations où $x_{\text{ExsA}} = 2$ (autrement dit, $inductible$ coïncide avec l'égalité $x_{\text{ExsA}} = 2$) ; en-dessous de cette valeur, la bactérie est non inductible (car nous considérons dans ce modèle simplifié que le signal d'activation (milieu pauvre en calcium) est toujours présent, l'état inductible est donc ici assimilé à celui d'induit). Le système doit donc vérifier la formule LTL φ suivante[6] (correspondant à $\varphi_1 \wedge \varphi_2$) : $\mathbf{G}(x_{\text{ExsA}} = 2 \Rightarrow \mathbf{FG}\; x_{\text{ToxSST3}} = 1) \wedge \mathbf{G}(x_{\text{ExsA}} < 2 \Rightarrow \mathbf{FG}\; x_{\text{ToxSST3}} = 0)$. □

Un automate de Büchi est un système de transition fini permettant de reconnaître des mots de longueur infinie. Nous souhaitons reconnaître des chemins extraits des dynamiques d'un GRN G. Cela explique que la définition des automates de Büchi ci-dessous porte par défaut sur des variables propositionnelles de la forme $x_i \bowtie c$ avec x_i le niveau d'expression du gène $i \in G$, $\bowtie \in \{=, \neq, <, >, \leq, \geq\}$ et $c \in \mathbb{N}$.

DÉFINITION 12 (Automate de Büchi). — *Un Automate de Büchi est un tuple $(Q, q_0, A, \delta, \Sigma)$ où Q est l'ensemble des états, $q_0 \in Q$ est l'état initial, $A \subseteq Q$ est l'ensemble des états acceptants, $\delta : Q \times \Sigma \rightarrow 2^Q$ est l'ensemble des transitions et Σ est l'alphabet défini par l'ensemble des formules propositionnelles sur les propositions atomiques de la forme $x_i \bowtie c$.*

Une séquence infinie d'états fournie avec des valeurs de vérité pour toutes les propositions atomiques $x_i \bowtie c$ est appelée chemin. Un chemin est accepté par l'automate s'il passe infiniment souvent par un état acceptant. Une formule LTL φ peut être traduite en un automate de Büchi $B(\varphi)$ tel qu'une séquence infinie d'états vérifie φ si et seulement si le chemin associé est accepté par l'automate $B(\varphi)$.

Nous souhaitons identifier les dynamiques où tous les chemins sont compatibles avec les propriétés biologiques. Pour cela, nous voulons exclure les dynamiques où au moins un chemin ne satisfait pas l'une des propriétés. Ainsi, nous construisons l'automate de Büchi associé à la *négation* d'une formule LTL : $B(\neg\varphi)$.

---

5. Remarquons que nous considérons que le caractère cytotoxique peut ne pas être acquis ou perdu immédiatement ; d'autres interprétations sont possibles.

6. Notons ici que nous nous éloignons de la formule utilisée par (Filopon *et al.*, 2006) car nous considérons explicitement que le signal est activé.

EXEMPLE 13. — L'automate de Büchi présenté en figure 4, noté $B_{\neg\varphi}$, reconnaît la négation de la formule φ introduite dans l'exemple 11. Il a été généré en utilisant l'outil LTL2BA4J (Bodden, 2011). □

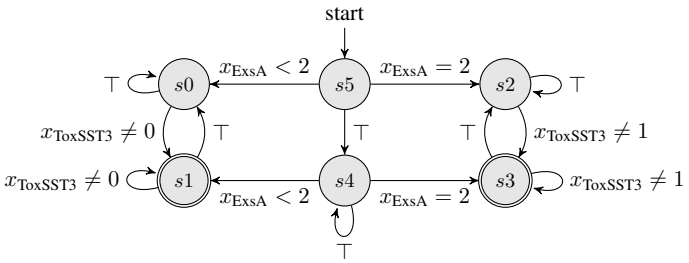


Figure 4. $B_{\neg\varphi} = B(\neg\varphi)$ *avec*
$\neg\varphi \equiv \mathbf{F}(x_{ExsA} = 2 \wedge \mathbf{GF}\ x_{ToxSST3} \neq 1) \vee \mathbf{F}(x_{ExsA} < 2 \wedge \mathbf{GF}\ x_{ToxSST3} \neq 0)$

À partir de l'automate de Büchi $B(\neg\varphi)$ et du PGRN P, nous construisons le Produit $\Pi = P \otimes B(\neg\varphi)$ pour vérifier le méta-modèle P au regard de φ.

DÉFINITION 14 (Produit). — *Soit* $P = (St, Tr)$ *un PGRN et* $B(\neg\varphi) = (Q_B, q_{0B}, A_B, \delta_B, \Sigma_B)$ *un automate de Büchi associé à la formule LTL* $\neg\varphi$*. Le Produit* $\Pi = P \otimes B(\neg\varphi)$ *est le tuple* $(Q_\Pi, q_{0\Pi}, A_\Pi, \delta_\Pi)$ *avec* $Q_\Pi = St \times Q_B$ *l'ensemble des sommets,* $q_{0\Pi} = (I, q_{0B})$ *le sommet initial,* $A_\Pi = St \times A_B$ *l'ensemble des sommets acceptants, et* δ_Π *l'ensemble des transitions de la forme* $((s, q_B), g \wedge g_B, a, (s', q'_B))$ *tel qu'il existe* $q'_B \in \delta_B(q_B, g_B)$*,* $q'_B \in Q_B$*, et* $(s, g, a, s') \in Tr$ *et tel que* $g \wedge g_B$ *est une formule satisfaisable* [7].

EXEMPLE 15 (Produit). — Le Produit Π_{1a} entre le PGRN de G_{1a} (figure 2) et l'automate de Büchi $B_{\neg\varphi}$ (figure 4) contient 12 sommets et 128 transitions (les transitions contenant des gardes avec des disjonctions sont séparées en plusieurs transitions de même source et même cible). □

3.2. Recherche des cycles paramétrés atteignables et acceptants

Le Produit obtenu à partir du PGRN est tel qu'un chemin du Produit correspond à la fois à un chemin du PGRN et à un chemin dans l'automate de Büchi. Ainsi, en considérant un jeu de valeurs particulier pour les paramètres biologiques, un chemin du Produit pour lequel le jeu de valeurs choisi satisfait les gardes des transitions franchies définit un chemin de la dynamique définie par le jeu de valeurs considéré. Si ce chemin passe infiniment souvent par un état acceptant, il correspond à un chemin qui satisfait la propriété LTL associée à l'automate de Büchi.

7. Une formule propositionnelle avec variables est satisfaisable s'il existe une substitution des variables rendant la formule vraie.

Nous nous appuyons sur des techniques d'exécution symbolique afin de ne pas évaluer les paramètres biologiques. Ces techniques ont été initialement développées pour le test de programmes (King, 1975). Il s'agit de manipuler symboliquement les variables au lieu de leur associer des valeurs concrètes. Chaque exécution symbolique est munie d'une *condition de chemin* exprimant les conditions nécessaires que doivent respecter les variables symboliques pour exécuter le chemin en question. Les techniques d'exécution symbolique ont été étendues aux systèmes de transition symbolique (Gaston *et al.*, 2006) en dépliant ces systèmes pour obtenir des arbres symboliques. Les chemins du système de transition étant infinis, les arbres symboliques sont eux-mêmes potentiellement infinis. Des critères de sélection ou repliements sont donc utilisés pour arrêter la construction des chemins, et rendre les arbres finis. Dans la suite, nous nous attacherons donc à identifier les situations de retour sur un nœud déjà rencontré qui autorisent à stopper l'exécution symbolique des chemins. Ces situations révèlent la présence de cycle ; si un cycle contient un état acceptant, alors il définit un chemin passant infiniment souvent par un état acceptant.

Le Produit, en tant que produit d'un automate de Büchi et d'un PGRN, inclut deux familles de variables : celles représentant les niveaux d'expression des gènes du GRN (les variables x_i) et celles représentant les paramètres biologiques (les variables $K_i(\omega)$). Ces deux ensembles de variables sont de nature différente. Les niveaux d'expression x_i sont *évalués* et *dynamiques*, leurs valeurs sont toutes toujours connues et elles sont susceptibles d'être modifiées après franchissement d'une transition. Les paramètres $K_i(\omega)$ sont *symboliques* et *statiques*, leurs valeurs sont inconnues mais elles peuvent (ou non) être contraintes au fur et à mesure du parcours du Produit, par l'ajout de *conditions de chemin* qui doivent rester compatibles entre elles.

La recherche de cycles acceptants s'effectue donc en exécutant symboliquement le Produit $\Pi = P \otimes B(\neg\varphi)$. Π est déplié, menant à la création de plusieurs *Arbres d'Exécution Symbolique* (*Symbolic Execution Trees* ou SET). Chaque SET est récursivement construit à partir d'un nœud de départ, appelé *racine*. Les nœuds d'un SET se composent d'un triplet (q, x, pc) où q est un sommet de Π, $x = (x_i)_{i\in V}$ est un état, et pc est la condition de chemin (ou *path condition*), autrement dit la conjonction des contraintes que doivent respecter les paramètres $K_i(\omega)$ pour s'assurer que le nœud est atteignable depuis la racine. Les racines quant à elles sont de la forme $(q_{0\Pi}, x, \top)$ où $q_{0\Pi}$ est le sommet initial de Π, x un état quelconque, et $\top$ la condition de chemin toujours vraie [8]. Il existe donc autant de SET que d'états différents (ce nombre pouvant cependant être réduit pour calquer une expérience biologique).

DÉFINITION 16 (Arbre d'exécution symbolique à partir d'un état). — *Soit* $\Pi = (Q_\Pi, q_{0\Pi}, A_\Pi, \delta_\Pi)$ *un Produit et* $x = (x_i)_{i\in V}$ *un état. L'arbre d'exécution symbolique associé à* Π *pour l'état* x *est le tuple* $\mathcal{S}(x) = (Q_S, \delta_S)$ *où* Q_S *est l'ensemble des nœuds et* δ_S *l'ensemble des transitions, mutuellement définis par :*

– $(q_{0\Pi}, x, \top) \in Q_S$ *est appelée racine de l'arbre,*

8. Nous considérons que les contraintes initiales sur les paramètres (définies en section 2.4) sont implicitement prises en compte.

– *pour tout* $(q^k, x^k, pc^k) \in Q_S$*, pour tout* $(q^k, g, a, q') \in \delta_\Pi$*, si* $pc^k \wedge g[x^k]$ *est satisfaisable (avec* $g[x^k]$ *la formule propositionnelle dérivée de* g *en substituant les variables* x_i *par la* i*ème composante de l'état* x^k*) alors* $(q', a(x^k), pc^k \wedge g[x^k]) \in Q_S$.

Les successeurs d'un nœud du SET sont construits en considérant toutes les transitions issues du sommet lui correspondant dans Π. Notons $\sigma^k = (q^k, x^k, pc^k)$ le $k^{\text{ème}}$ nœud du chemin $\sigma^1, \ldots, \sigma^k$ en cours de parcours. Soit $(q^k, g, a, q') \in \delta_\Pi$ une transition de Π issue de q^k. Cette transition peut être franchie si $pc^k \wedge g[x^k]$ est satisfaisable. Un nouveau nœud $\sigma^{k+1} = (q^{k+1}, x^{k+1}, pc^{k+1})$ est alors créé en mettant à jour l'état en fonction de l'affectation a, et en complétant la condition de chemin par la garde : $q^{k+1} = q'$, $x^{k+1} = a(x^k)$ et $pc^{k+1} = pc^k \wedge g[x^k]$.

Par construction, la condition de chemin d'un nœud est constituée de la conjonction de toutes les gardes des transitions franchies par ses ascendants. Ainsi, les conditions de chemin augmentent avec l'allongement des chemins du SET, ce qui réduit le nombre de dynamiques compatibles avec le chemin en question. Remarquons que les jeux de paramètres qui satisfont pc^{k+1} satisfont également pc^k. Ainsi, la condition de chemin d'un nœud *implique* chacune des conditions de chemin de ses ascendants.

Les nœuds d'un SET sont de la forme (q, x, pc), et donc, abstraction faite des conditions de chemin pc, le nombre [9] possible de nœuds différents dans un SET est fini. Ainsi, lors de la construction d'un nouveau nœud $\sigma^k = (q^k, x^k, pc^k)$, si un nœud $\sigma^l = (q^k, x^k, pc^l)$, appelé *nœud de retour*, existe parmi ses ancêtres (*i.e.* les nœuds entre la racine et σ^k) alors nous pouvons arrêter l'analyse de ce chemin. En effet, puisque σ^k est un descendant de σ^l, l'ensemble des jeux de paramètres vérifiant pc^k est inclus dans l'ensemble des jeux de paramètres vérifiant pc^l. L'ensemble des transitions franchissables à partir de σ^k est donc un sous-ensemble des transitions franchissables à partir de σ^l.

De cette manière, en réalisant une exécution mêlant variables symboliques (les $K_i(\omega)$) et numériques (les x_i), nous pouvons arrêter le parcours du Produit Π, de façon à ce que chaque chemin des SET résultants soit fini et contienne un cycle (démarrant avec le nœud de retour σ^l et finissant avec la transition menant à σ^k). Enfin, s'il existe un nœud acceptant entre le nœud de retour σ^l et le nœud σ^k, alors la condition de chemin de σ^k est dite *acceptante*.

EXEMPLE 17 (Cycle acceptant). — La figure 5 présente un exemple de détection d'une condition de chemin acceptante dans une branche d'un des arbres d'exécution symbolique construit à partir du Produit Π_{1a}. Le dernier nœud créé sur cette branche (cible de la transition 8) est dans le même état $((0, 0, 0))$ et lié au même sommet (Fs3) qu'un de ses ancêtres (cible de la transition 7), il y a donc un cycle. Comme Fs3 est un sommet acceptant, ce cycle est acceptant et la condition de chemin du nœud enfant

9. Ce nombre est majoré par le produit du nombre d'états possibles et du nombre de sommets du Produit ; ce dernier est lui-même majoré par le produit du nombre de sommets de l'automate de Büchi et du nombre de sommets (2) du PGRN.

est donc acceptante. Un autre cycle avait précédemment été détecté, entre les nœuds cibles des transitions 1 et 2 mais ne contenait pas de sommets acceptants. □

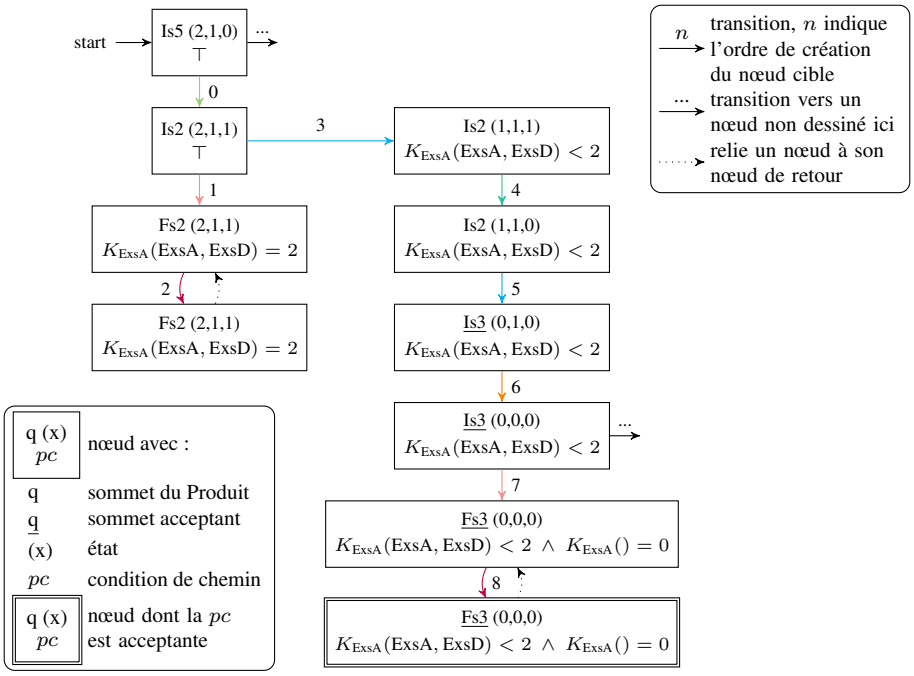


Figure 5. Cycles acceptant et non acceptant

L'exécution symbolique du Produit $\Pi = P \otimes B(\neg\varphi)$ nous conduit à construire un ensemble de SET $\{\mathcal{S}(x)|x = (x_i)_{i\in V}\}$. Un arbre $\mathcal{S}(x)$ représente l'ensemble des chemins possibles à partir de l'état x. Ainsi, si un jeu de paramètres $K_i(\omega)$ vérifie (au moins) une des conditions de chemin acceptantes de $\mathcal{S}(x)$, cela signifie qu'il existe un chemin dans Π commençant en x et passant infiniment souvent par le cycle associé à cette condition de chemin acceptante. Autrement dit, le jeu de paramètres $K_i(\omega)$ définit une dynamique où au moins un des chemins commençant en x vérifie la propriété $\neg\varphi$.

Réciproquement, si le jeu de paramètres $K_i(\omega)$ ne vérifie aucune des conditions de chemins acceptantes, alors il n'existe pas de chemin partant de x vérifiant $\neg\varphi$, donc tous les chemins partant de x vérifient φ. Si on cible toutes les dynamiques qui vérifient φ (i.e. les dynamiques telles que tous les chemins, quel que soit l'état de départ, vérifient φ), il suffit de rechercher les jeux de paramètres vérifiant la *conjonction des négations des conditions de chemin acceptantes de tous les SET associés à* Π.

3.3. Réduction du parcours par coupures des branches

La notion de coupure recouvre l'identification de nœuds du SET dont l'analyse est inutile car les chemins vers lesquels ils pourraient mener ont déjà été analysés (dans une autre branche du même SET ou d'un autre SET précédemment construit). On peut alors « couper » la branche de l'arbre correspondant à ce nœud sans perte de solution.

Considérons deux nœuds $\sigma^k = (q^k, x^k, pc^k)$ et $\sigma^l = (q^k, x^k, pc^l)$ *sans lien d'ascendance* mais tels que $pc^k \Rightarrow pc^l$. Autrement dit, les jeux de paramètres qui vérifient pc^k vérifient aussi pc^l. Alors, comme pour le repliement, l'ensemble des transitions franchissables à partir de σ^k est un sous-ensemble des transitions franchissables à partir de σ^l. Si σ^l (et ses descendants) ont déjà été analysés alors il n'est pas nécessaire d'analyser σ^k. Dans pareil cas, on dira que σ^k est un nœud de coupure *inclus* dans σ^l.

EXEMPLE 18 (Coupure par inclusion de nœud). — Une autre branche d'un des arbres issus du Produit Π_{1a} est représentée en figure 6. Le nœud cible de la transition 6 est un nœud de coupure inclus dans le nœud cible de la transition 3 : ils n'ont pas de lien d'ascendance, ont le même sommet (Is0) et le même état $((0, 0, 0))$ et sa condition de chemin implique celle de l'autre nœud ($\top \Rightarrow \top$). □

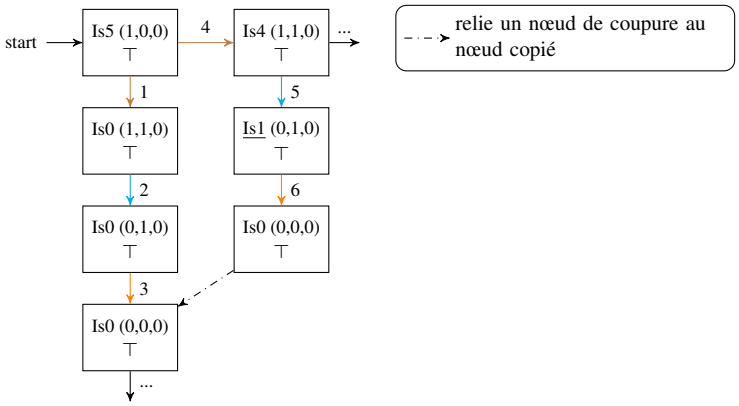


Figure 6. Coupure par inclusion de nœud

Il est également possible d'utiliser les conditions de chemins acceptantes déjà rencontrées pour couper certains nœuds. Notons $\{apc_1, \ldots, apc_l\}$ l'ensemble des conditions de chemins acceptantes connues et supposons que le nœud à créer $\sigma^k = (q^k, x^k, pc^k)$ soit tel que pc^k implique l'une des conditions de chemins acceptantes connues, notée apc. Alors, toutes les conditions de chemin des nœuds construits à partir de σ^k impliqueront pc^k (puisqu'ils descendent de σ^k) et donc apc.

EXEMPLE 19 (Coupure par implication d'une condition de chemin déjà acceptée). — La figure 7 représente une autre branche d'un des arbres du Produit Π_{1a}. Les

conditions de chemin des nœuds cibles des transitions 8 et 11, respectivement $K_{\text{ExsA}}(\text{ExsA}, \text{ExsD}) < 2 \wedge K_{\text{ExsA}}() > 0$ et $K_{\text{ExsA}}(\text{ExsA}, \text{ExsD}) < 2 \wedge K_{\text{ExsA}}() = 0$, sont acceptantes. Tous les nœuds pouvant être créés après eux avec une condition de chemin impliquant l'une de ces deux conditions (*i.e.* impliquant $K_{\text{ExsA}}(\text{ExsA}, \text{ExsD}) < 2$) sont donc inutiles. Comme cette condition a été ajoutée à partir du nœud cible de la transition 3, nous pouvons couper directement toutes les sous-branches de ce nœud et recommencer le parcours à partir de son père (cible de la transition 0). Nous voyons ainsi que le prochain nœud créé (cible de la transition 12) est un fils du premier nœud créé. □

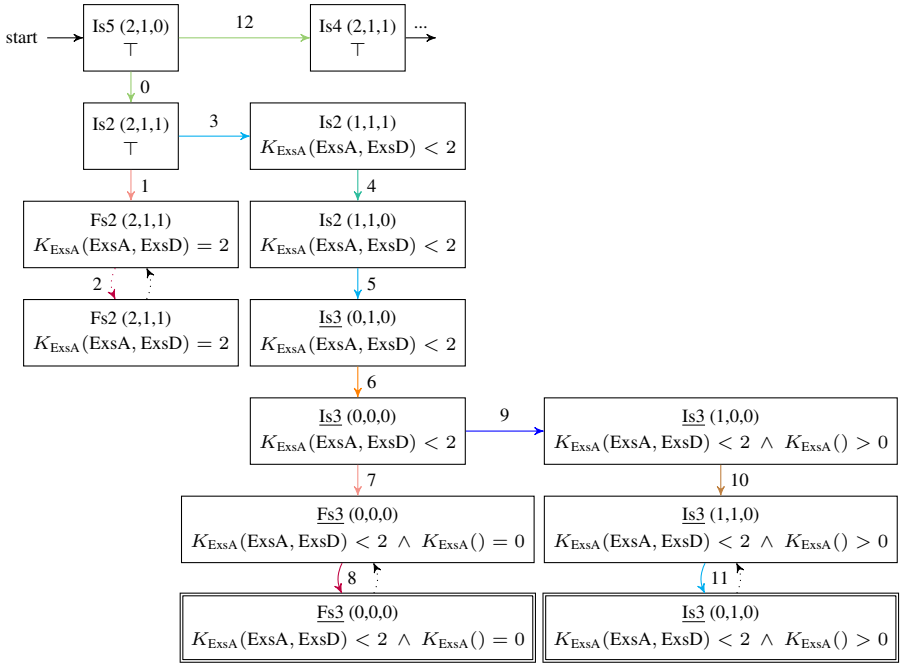


Figure 7. Coupures par implication de conditions de chemin acceptantes

EXEMPLE 20. — Six arbres d'exécution symbolique (correspondant aux six [10] états possibles) peuvent être construits à partir du Produit Π_{1a}. Les figures 5 et 7, et la figure 6 représentent des extraits des deux premiers SET construits, de racines respectives $(\text{Is5}, (2,1,0), \top)$ et $(\text{Is5}, (1,1,0), \top)$. Notons que les deux premiers SET construits possèdent respectivement dans leur totalité 22 et 24 nœuds, alors que les autres arbres en possèdent respectivement 13, 3, 3 et 3. Cette diminution de la taille des

10. Nous rappelons que les seuls états initiaux du système sont les états sans toxine, soit $x_{\text{ToxSST3}} = 0$.

arbres est représentative de ce que l'on observe avec d'autres exemples ; en effet, les arbres suivants vont avoir tendance à réutiliser l'exploration des arbres précédents, en coupant les branches dont les conditions sur le chemin correspondent à des conditions de chemin acceptantes et en produisant de plus en plus de nœuds de coupure.

Les nœuds acceptants sont au nombre de trois, leurs conditions de chemin sont :
$K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\}) < 2 \ \wedge \ K_{\text{ExsA}}(\{\}) = 0$,
$K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\}) < 2 \ \wedge K_{\text{ExsA}}(\{\}) > 0$,
et $K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\}) = 2 \ \wedge K_{\text{ExsA}}(\{\}) = 2$.
La conjonction des contraintes initiales sur les paramètres (définition 9) et des négations des conditions de chemin acceptantes, nous permet d'obtenir une seule solution, définie par les valeurs suivantes : $K_{\text{ExsA}}(\{\}) = 1$, $K_{\text{ExsA}}(\{\text{ExsA}\}) = 2$, $K_{\text{ExsA}}(\{\text{ExsD}\}) = 0$, $K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\}) = 2$, $K_{\text{ExsD}}(\{\}) = 0$, $K_{\text{ExsD}}(\{\text{ExsA}\}) = 1$, $K_{\text{ToxSST3}}(\{\}) = 0$ et $K_{\text{ToxSST3}}(\{\text{ExsA}\}) = 1$. La dynamique correspondante est présentée en figure 8.

En appliquant notre méthode sur le graphe d'interaction G_{1b} (figure 1b, 2ème variante du graphe d'interaction du même GRN), nous n'obtenons pas de solution : un seul graphe d'interaction est donc compatible avec la propriété biologique testée. Remarquons que nous divergeons ici du résultat obtenu par (Filopon *et al.*, 2006), *i.e.* une solution au graphe d'interaction G_{1b}. Ceci est causé par le fait que nous exprimons différemment les propriétés biologiques en LTL et que nous étudions la variation de x_{ToxSST3} et non pas seulement de x_{ExsA} et x_{ExsD} (l'instanciation des paramètres qu'ils proposent peut conduire à un cycle infini d'augmentation/diminution de x_{ExsA} avec possibilité – mais pas obligation – d'augmentation de x_{ToxSST3}). Insistons ici sur le fait que cette différence n'est pas due à l'outil utilisé (SPuTNIk/SMBioNet) mais uniquement aux formules LTL données en entrée ; en testant les formules de (Filopon *et al.*, 2006) avec SPuTNIk, nous retrouvons bien la solution trouvée avec SMBioNet. □

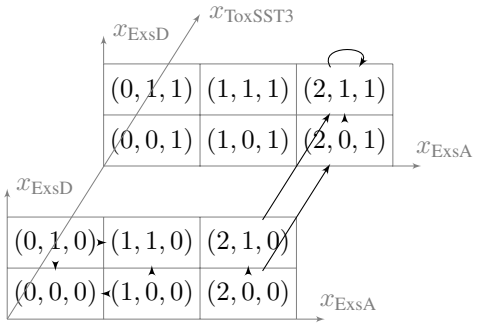


Figure 8. Dynamique associée à G_{1a} vérifiant φ dans un système initialement exempt de toxines

3.4. Algorithme de parcours des SET

L'algorithme 1 décrit la mise en œuvre de la recherche des conditions de chemin acceptantes ; il s'appuie sur un *parcours en profondeur*. Trois variables globales sont utilisées : le Produit Π, la liste $acceptingPC$ des conditions de chemin acceptantes et la liste $nodesList$ des nœuds du SET en cours de construction. Le premier appel à la fonction DFS est réalisé à partir d'un nœud racine. Les lignes 1 à 3 de l'algorithme permettent le calcul des successeurs d'un nœud du SET comme expliqué dans la partie 3.2. Rappelons que nous ne parcourons que les nœuds dont la condition de chemin satisfait les contraintes sur les paramètres (définies en section 2.4). Trois tests sont alors réalisés successivement, correspondant aux repliements et coupures précédemment décrits :

– La ligne 4 teste si la condition de chemin du successeur implique l'une des conditions de chemin acceptantes de la liste $acceptingPC$ (deuxième type de coupure décrit dans la section 3.3). En cas de réponse positive, l'étude de ce successeur n'est pas poursuivie et l'itération suivante de la boucle *pour chaque* teste une autre transition partant du sommet du nœud courant.

– En cas de réponse négative à ce test, la ligne 5 vérifie l'existence d'un cycle, commençant avec un ancêtre du successeur testé et qui serait dans le même état et lié au même sommet de Π. Si un tel ancêtre est trouvé et qu'un nœud acceptant est présent dans ce cycle, alors la condition de chemin du successeur est acceptante et est ajoutée à la liste $acceptingPC$ (lignes 6 à 7).

– Si aucun des deux tests précédents n'est positif, la ligne 8 vérifie si le successeur est inclus dans un nœud déjà créé (présent dans $nodesList$), ce qui correspond au premier type de coupure de la section 3.3. Si un tel nœud existe déjà, l'algorithme passe au successeur suivant ; sinon, la fonction DFS est rappelée avec le successeur en argument (ligne 9) et celui-ci est ensuite ajouté à $nodesList$ (ligne 10).

Algorithme 1 : Vue d'ensemble de DFS($node$)

Données : $node$, global Π, global $acceptingPC$, global $nodesList$

1 **pour chaque** $transition$ *de* Π *en sortie du sommet de* $node$ **faire**
2 **si** $transition$ *est* SATISFAISABLE **alors**
3 CALCULER $succ$, le nœud successeur atteint par cette transition;
4 **si** *la* pc *de* $succ$ *n'*IMPLIQUE *pas l'une des* pc *dans* $acceptingPC$ **alors**
5 **si** $succ$ *est un* NŒUD DE RETOUR *vers un ancêtre* anc **alors**
6 **si** *il existe un* NŒUD ACCEPTANT *entre* anc *et* $succ$ **alors**
7 Ajouter la pc de $succ$ à $acceptingPC$;
8 **sinon si** $succ$ *n'est pas* INCLUS *dans un nœud de* $nodesList$ **alors**
9 DFS($succ$);
10 Ajouter $succ$ à $nodesList$;

Quand l'exploration de tous les SET est terminée, nous procédons à la conjonction des contraintes sur les paramètres et des négations de chacune des conditions de

chemin acceptantes stockées dans $acceptingPC$. Les jeux de paramètres qui vérifient cette conjonction définissent les dynamiques qui satisfont la propriété biologique.

3.5. SPuTNIk

Nous avons développé un prototype, appelé SPuTNIk [11], pour déterminer les jeux de paramètres biologiques d'un réseau de régulation biologique compatibles avec des connaissances biologiques, conformément à la méthode précédemment décrite. SPuTNIk prend en entrée un graphe d'interaction ainsi qu'une ou plusieurs formules LTL à tester (via une interface graphique ou un fichier donné en argument d'une ligne de commande). En sortie, l'utilisateur obtient la liste des jeux de paramètres satisfaisant la(les) formule(s) LTL en entrée.

SPuTNIk est écrit en Java. Nous utilisons le solveur de contrainte Z3 (De Moura, Bjørner, 2008) afin de vérifier la satisfaisabilité des conditions de chemin. D'un point de vue technique, nous mettons à profit le système de pile du solveur, qui permet d'ajouter la garde de chaque transition franchie pendant le parcours en profondeur puis de la retirer en remontant (après traitement d'un nœud et de ses successeurs), afin d'emprunter une autre branche du SET [12]. Nous utilisons également Z3 en phase finale afin de déterminer les jeux de paramètre solutions. D'autre part, nous faisons appel à la bibliothèque LTL2BA4J (Bodden, 2011) (qui est une passerelle vers la bibliothèque ltl2ba (Gastin, Oddoux, 2001)) pour générer un automate de Büchi à partir d'une formule LTL.

À titre indicatif, nous donnons dans le tableau 1 les temps d'exécution [13] pour plusieurs exemples de formules LTL pour le graphe d'interaction G_{1a}. Notons que les temps sont très variables selon la formule LTL à tester (car la taille de l'automate de Büchi, et donc du Produit à parcourir, dépend de la formule).

Tableau 1. Temps d'exécution pour plusieurs formules LTL appliquées à G_{1a}

Formule LTL (avec la convention A : sur tous les chemins et E : sur au moins un chemin)	Moyenne sur 100 exécutions
Sans formule (énumération directe des solutions)	344 ms
$A : \mathbf{G}(x_{\text{ExsA}} = 2 \Rightarrow \mathbf{FG}\, x_{\text{ToxSST3}} = 1) \wedge \mathbf{G}(x_{\text{ExsA}} < 2 \Rightarrow \mathbf{FG}\, x_{\text{ToxSST3}} = 0)$	484 ms
$E : \mathbf{F}(x_{\text{ExsA}} = 2 \Rightarrow \mathbf{GF}\, x_{\text{ToxSST3}} = 0) \vee \mathbf{F}(x_{\text{ExsA}} < 2 \Rightarrow \mathbf{GF}\, x_{\text{ToxSST3}} = 1)$	416 ms
$E : e1 \wedge \mathbf{F}(e2 \wedge \mathbf{F}(e3 \wedge \mathbf{F}(e4)))$	504 ms
avec $e1 \equiv x_{\text{ExsA}} = 2 \wedge x_{\text{ExsD}} = 1 \wedge x_{\text{ToxSST3}} = 0$, $e2 \equiv x_{\text{ExsA}} = 1 \wedge x_{\text{ExsD}} = 1 \wedge x_{\text{ToxSST3}} = 1$, $e3 \equiv x_{\text{ExsA}} = 0 \wedge x_{\text{ExsD}} = 0 \wedge x_{\text{ToxSST3}} = 0$, $e4 \equiv x_{\text{ExsA}} = 1 \wedge x_{\text{ExsD}} = 1 \wedge x_{\text{ToxSST3}} = 0$	

Dans (Gallet *et al.*, 2014b) nous avons traité un autre cas d'étude, portant sur le switch lytique/lysogénique du phage λ, dont le GRN est composé de 4 gènes et 10

11. *Symbolic Parameters of Thomas' Networks Inference*

12. Les contraintes initiales sur les paramètres, si requises, sont initialement ajoutées à la base de la pile de manière à ne jamais les supprimer.

13. sur un i7-3520M 4 cœurs à 2.90GHz sous Linux

interactions. Les formules LTL testées sont issues de (Klarner *et al.*, 2012) ; elles étaient initialement sous la forme de deux *séries temporelles*, définies comme des séquences d'états $s_1, \ldots, s_k$ à atteindre. L'outil Parsybone (Streck, 2014) est optimisé pour traiter des séries temporelles, il traite ces deux séries en [14] 22 s. Chaque série peut être exprimée sous la forme d'une formule LTL avec l'opérateur $\mathbf{F}$ imbriqué, i.e. sous la forme $\mathbf{F}(s_1 \wedge \mathbf{F}(s_2 \ldots \wedge \mathbf{F}(s_k)))$. Avec SPuTNIk, nous obtenons un temps d'exécution de 3 min 51 s (dont 34 s consacrées à l'énumération des solutions, très lente avec Z3). Parsybone peut également prendre en entrée un automate de Büchi, mais est plus lent dans ce cas. En lui fournissant les deux automates de Büchi correspondant aux deux séries temporelles, il a alors un temps d'exécution [13] de 9 min.

4. Inductibilité de la cytotoxicité de *P. aeruginosa*

4.1. Interprétation du résultat du GRN minimal

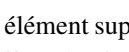
Au fil de cet article, nous avons étudié un GRN composé de trois gènes mis en cause dans l'inductibilité de la cytotoxicité de *P. aeruginosa*. Grâce à SPuTNIk, nous avons déterminé qu'une version du graphe d'interaction admet une solution satisfaisant l'hypothèse du switch épigénétique du SST3 de la bactérie. L'augmentation (resp. diminution) de ExsA modifie le phénotype de la bactérie, d'un état non-cytotoxique (resp. cytotoxique) à un état cytotoxique (resp. non cytotoxique) ; ces deux états sont stables. D'autre part, la concentration de l'inhibiteur de ExsA, ExsD, ne semble pas intervenir dans le phénomène. Dans (Filopon *et al.*, 2006), les auteurs obtiennent le même résultat avec leur outil SMBioNet (à la différence du résultat obtenu avec le deuxième graphe d'interaction, comme expliqué précédemment dans l'exemple 20). Ce résultat amène une idée du protocole expérimental à entreprendre pour tester l'hypothèse du switch épigénétique. Ainsi, (Filopon *et al.*, 2006) ont testé les conséquences d'un pulse de ExsA (pour l'amener à saturation, *i.e.* au-dessus du seuil de concentration représenté discrètement par $x_{\text{ExsA}} = 2$) *in vitro* pour une souche de *P. aeruginosa* non inductible et soumise à une déplétion calcique. Ils ont constaté que la souche produisait des toxines sur plusieurs générations (environ dix). Ils supposent donc que ce phénotype acquis est stable, ce qui confirme l'hypothèse épigénétique.

4.2. GRN étendu

Le GRN étudié précédemment (figure 1) est une version réduite à l'essentiel du GRN en cause dans l'inductibilité de la cytotoxicité de *P. aeruginosa*. De façon plus précise, ExsD inhibe l'activité de ExsA empêchant la production et la sécrétion des toxines. Mais les protéines synthétisées à partir d'un autre gène, ExsC, peuvent elles-mêmes se lier à ExsD, empêchant la formation du complexe ExsD-ExsA, qui est à l'origine de l'inhibition de ExsA par ExsD. Or ExsC n'est libre de se fixer à ExsD qu'à la condition qu'elle ne fasse pas elle-même partie d'un complexe avec les protéines

14. Exécuté sur leur serveur, dont les caractéristiques sont inconnues (temps d'exécution donné par l'outil).

d'un autre gène, ExsE. Cependant, pendant la sécrétion, ExsE est exportée de la bactérie à travers le SST3, libérant ainsi ExsC ; par réaction en chaîne, ExsA est alors libérée de l'inhibition de ExsD et active la production de toxines.

Pour prendre en compte plus précisément les mécanismes liés à l'appauvrissement du milieu en calcium, nous ajoutons ainsi les gènes ExsC et ExsE dans le graphe d'interaction. La formation des complexes est représentée via des interactions négatives. De plus, nous symbolisons l'influence de la déplétion calcique dans le graphe d'interaction sous la forme d'un élément supplémentaire, noté : 

(et Ca\ dans le texte). Pour distinguer ses effets des interactions entre les gènes, nous utilisons la représentation graphique ⇢. Enfin, nous séparons le complexe représentant les gènes producteurs des toxines de ceux codant l'appareil de sécrétion (anciennement ToxSST3, respectivement séparé en Tox et SST3). Ces deux complexes sont en effet régulés par ExsA mais a priori à un niveau de concentration différent.

Le graphe d'interaction obtenu est présenté en figure 9. Les valeurs des seuils de régulation correspondent au modèle biologique le plus simple, selon le principe du rasoir d'Ockham. Nous avons ainsi fixé à la même valeur les seuils de régulation des gènes se trouvant sur le même opéron [15], soit : $T(\text{ExsA}, \text{ExsA}) = T(\text{ExsA}, \text{ExsC}) = T(\text{ExsA}, \text{ExsE})$ et $T(\text{ExsA}, \text{SST3}) = T(\text{ExsA}, \text{ExsD})$. D'autre part, nous avons limité ExsA à quatre états différents, correspondant directement aux états biologiques *non exprimé*, *non inductible*, *inductible mais non induit* et *induit*, avec *induit* correspondant à la valeur maximale que peut atteindre ExsA (soit la valeur du seuil $T(\text{ExsA}, \text{Tox})$). Après application des contraintes sur les paramètres (décrites en section 2.4), il reste 16384 jeux de paramètres différents possibles.

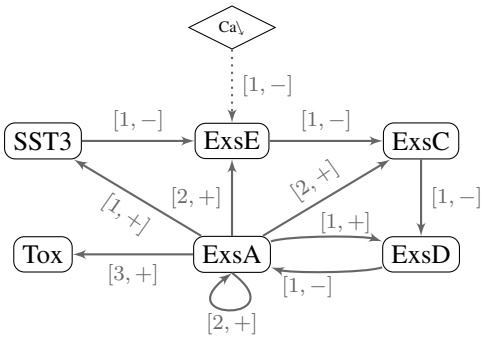


Figure 9. Graphe d'interaction du GRN étendu

Nous réutilisons les propriétés φ_1 et φ_2 (avec cependant une légère adaptation de la traduction en LTL), déjà illustrées sur le GRN minimal, et ajoutons une pro-

15. Un opéron est une unité d'ADN regroupant un ensemble de gènes sous influence du même régulateur, en général transcrits à partir d'un seuil de régulation commun.

priété, φ_3, représentant l'effet de l'absence de déplétion calcique, définie ainsi : $\varphi_3 \equiv \mathbf{G}\big((inductible \wedge \neg depletionCalcium) \Rightarrow \mathbf{FG}\,\neg cytotoxique\big).\, depletionCalcium$ correspond à un milieu appauvri en calcium, soit $\text{Ca}\backslash_{\searrow} = 1$.

Dans ce système, deux niveaux d'expression du gène ExsA représentent les états inductibles : si $x_{\text{ExsA}} = 3$ alors les toxines sont produites, *i.e.* le système est dans un état induit, si $x_{\text{ExsA}} = 2$ alors le système est dans un état inductible mais non induit. Ainsi, les propriété biologiques φ_1 et φ_2 doivent être réécrites de la manière suivante :
$\varphi_1 \equiv \mathbf{G}\big((x_{\text{ExsA}} \geq 2 \wedge \text{Ca}\backslash_{\searrow} = 1) \Rightarrow \mathbf{FG}\, x_{\text{Tox}} = 1\big)$,
$\varphi_2 \equiv \mathbf{G}(x_{\text{ExsA}} < 2 \Rightarrow \mathbf{FG}\, x_{\text{Tox}} = 0)$
et $\varphi_3 \equiv \mathbf{G}\big((x_{\text{ExsA}} \geq 2 \wedge \text{Ca}\backslash_{\searrow} = 0) \Rightarrow \mathbf{FG}\, x_{\text{Tox}} = 0\big)$.

Nous avons par ailleurs des connaissances biologiques liées à la formation des complexes inhibiteurs, s'appliquant directement aux paramètres afin de limiter les valeurs qu'ils peuvent prendre. Si ExsD inhibe ExsA alors il n'y a pas production de toxines donc, dans cette configuration, son niveau d'expression doit être strictement inférieur au seuil de production des toxines, soit : $K_{\text{ExsA}}(\{\text{ExsA}, \text{ExsD}\}) \leq 2$. ExsC et ExsD peuvent également former un complexe, signifiant que l'inhibition de ExsC sur ExsD est plus forte que l'activation de ExsA sur ExsD, soit : $K_{\text{ExsD}}(\{\text{ExsA}, \text{ExsC}\}) = 0$. De la même façon, l'inhibition de ExsE sur ExsC est plus forte que l'activation de ExsA sur ExsC, soit : $K_{\text{ExsC}}(\{\text{ExsA}, \text{ExsE}\}) = 0$.

D'autre part, nous savons également que l'effet de $\text{Ca}\backslash_{\searrow}$ et de SST3 sur ExsE est conjoint (car ExsE n'est exportée qu'à condition que l'appareil de sécrétion soit formé et sous réserve que le signal d'activation soit donné), aboutissant aux égalités suivantes : $K_{\text{ExsE}}(\{\}) = K_{\text{ExsE}}(\{\text{Ca}\backslash_{\searrow}\}) = K_{\text{ExsE}}(\{\text{SST3}\})$ et $K_{\text{ExsE}}(\{\text{ExsA}\}) = K_{\text{ExsE}}(\{\text{Ca}\backslash_{\searrow}, \text{ExsA}\}) = K_{\text{ExsE}}(\{\text{ExsA}, \text{SST3}\})$.

En donnant en entrée à SPuTNIk le graphe d'interaction étendu et les trois formules LTL, nous obtenons cinq jeux de paramètres compatibles ; en ajoutant les contraintes supplémentaires sur les paramètres, une seule solution est conservée. Ainsi, le GRN étendu, qui prend en compte la présence de calcium dans le milieu ainsi que son appauvrissement de manière plus fine que le GRN réduit, est compatible avec l'hypothèse et les expériences sur le switch épigénétique du SST3.

4.3. Variations avec modifications des seuils des interactions et des propriétés LTL

Les seuils du GRN étendu ont été fixés en considérant que le modèle le plus simple devait être le plus probable. Cependant, en l'absence d'expériences biologiques corroborant ces valeurs, d'autres possibilités peuvent être envisagées. Nous avons ainsi testé avec SPuTNIk cinq autres configurations à partir du graphe d'interaction du GRN étendu, en conservant les égalités des seuils des régulations des gènes appartenant au même opéron et en gardant le seuil $T(\text{ExsA}, \text{Tox})$ maximal (en veillant à adapter les formules LTL en fonction de ce seuil).

Un seul des cinq modèles donne au moins une solution (une solution exactement). Il s'agit du graphe d'interaction dérivé du graphe présenté en figure 9 en apportant les

modifications suivantes : $T(\text{ExsA}, \text{ExsA}) = T(\text{ExsA}, \text{ExsC}) = T(\text{ExsA}, \text{ExsE}) = 1$ et $T(\text{ExsA}, \text{Tox}) = 2$. Ce modèle est cependant moins probable que le premier (en figure 9) car ExsA, ExsC, ExsE d'une part et ExsD, SST3 d'autre part appartiennent à deux opérons différents, il est donc peu probable que les seuils de ces deux groupes de gènes régulés par ExsA soient identiques.

Lors de l'étude du GRN minimal, nous avons vu que nous obtenions des résultats différents de (Filopon *et al.*, 2006) pour le graphe d'interaction G_{1b}, dus à une traduction différente des propriétés biologiques en LTL. Nous proposons à présent de faire un comparatif en considérant le GRN étendu.

L'adaptation des formules de (Filopon *et al.*, 2006) au GRN étendu nous donne :
$\varphi'_1 \equiv \mathbf{G}\big((inductible \wedge depletionCalcium) \Rightarrow \mathbf{XF}\, induit\big)$
$\varphi'_2 \equiv \mathbf{G}(\neg inductible \Rightarrow \neg \mathbf{F}\, inductible)$
$\varphi'_3 \equiv \mathbf{G}\big((inductible \wedge \neg depletionCalcium) \Rightarrow \mathbf{XF}\, \neg induit\big)$
avec $induit \equiv \text{ExsA} = 3$. Soit (pour le graphe d'interaction en figure 9) :
$\varphi'_1 \equiv \mathbf{G}\big((x_{\text{ExsA}} \geq 2 \wedge \text{Ca}\backslash_{\searrow} = 1) \Rightarrow \mathbf{XF}\, \text{ExsA} = 3\big)$
$\varphi'_2 \equiv \mathbf{G}(\neg x_{\text{ExsA}} \geq 2 \Rightarrow \neg \mathbf{F}\, x_{\text{ExsA}} \geq 2)$
$\varphi'_3 \equiv \mathbf{G}\big((x_{\text{ExsA}} \geq 2 \wedge \text{Ca}\backslash_{\searrow} = 0) \Rightarrow \mathbf{XF}\, \text{ExsA} < 3\big)$.

SPUTNIK donne deux solutions avec ces formules, y compris la solution obtenue précédemment avec $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$. Nous avons également testé les graphes d'interaction modifiés présentés ci-dessus : nous obtenons une solution avec le modèle tel que $T(\text{ExsA}, \text{ExsA}) = T(\text{ExsA}, \text{ExsC}) = T(\text{ExsA}, \text{ExsE}) = 1$ et $T(\text{ExsA}, \text{Tox}) = 2$ et 2 solutions avec un autre modèle (qui n'obtenait pas de solution avec les formules précédentes) obtenu à partir du graphe en figure 9 en apportant les modifications suivantes : $T(\text{ExsA}, \text{ExsA}) = T(\text{ExsA}, \text{ExsC}) = T(\text{ExsA}, \text{ExsE}) = 1$, $T(\text{ExsA}, \text{SST3}) = T(\text{ExsA}, \text{ExsD}) = 2$ et $T(\text{ExsA}, \text{Tox}) = 2$.
Une modification légère dans l'interprétation des propriétés biologiques est donc susceptible de valider ou d'invalider un modèle.

5. Conclusion

Dans cet article, nous avons présenté une méthode de rétro-ingénierie des dynamiques de réseaux de régulation génétique. Nous avons encodé l'ensemble des dynamiques d'un GRN dans une représentation compacte, appelé GRN paramétré, manipulant les paramètres biologiques sous forme de variables. Les connaissances biologiques sont modélisées à l'aide de formules en langage LTL. Notre algorithme d'inférence de paramètres s'applique à un GRN Paramétré et à une formule LTL en s'inspirant des algorithmes de model checking LTL de recherche de cycles acceptants. Les jeux de paramètres définissant des dynamiques en accord avec la formule LTL sont solutions de contraintes caractéristiques de la présence de cycles acceptants dans le dépliage contrôlé de la structure produite. Cette analyse est implémentée à l'aide de notre prototype SPUTNIK et nous avons illustré ses performances à l'aide d'une étude de cas, celui de l'inductibilité de la cytotoxicité de *P. aeruginosa*.

Nos perspectives de travail portent sur l'amélioration de notre prototype SPuTNIk, en terme de temps d'exécution et d'occupation mémoire. Nous envisageons plusieurs pistes : la parallélisation de l'exécution symbolique des SET (en découpant le Produit en plusieurs composantes fortement connexes) ; la mise en place d'heuristiques pour les coupures ; l'emploi d'un solveur autre que Z3 pour l'énumération des solutions.

Remerciements

Les auteurs tiennent à remercier chaleureusement Janine Guespin-Michel pour les nombreuses discussions à propos de l'étude de cas P. aeruginosa*. Ils ont particulièrement apprécié sa disponibilité, sa gentillesse et son expertise de biologiste.*

Bibliographie

Baier C., Katoen J.-P. (2008). *Principles of Model Checking* (vol. 26202649). The MIT Press.

Barnat J., Brim L., Krejci A., Streck A., Safránek D., Vejnar M. *et al*. (2012). On parameter synthesis by parallel model checking. *IEEE/ACM Trans. Comput. Biology Bioinform.*, vol. 9, n° 3, p. 693-705.

Bernot G., Comet J.-P., Richard A., Guespin J. (2004, août). Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, vol. 229, n° 3, p. 339–347.

Bodden E. (2011). LTL2BA4J software, http://www.sable.mcgill.ca/~ebodde/rv/ltl2ba4j/ Manuel de logiciel. RWTH Aachen University.

Corblin F., Fanchon E., Trilling L. (2010). Applications of a formal approach to decipher discrete genetic networks. *BMC Bioinformatics*, vol. 11, p. 385.

Corblin F., Tripodi S., Fanchon E., Ropers D., Trilling L. (2009). A declarative constraint-based method for analyzing discrete genetic regulatory networks. *BioSystems*, vol. 98, p. 91-104.

De Moura L., Bjørner N. (2008). Z3: An efficient SMT solver. In *Tools and algorithms for the construction and analysis of systems*, vol. 4963, p. 337–340. Springer.

Filopon D., Mérieau A., Bernot G., Comet J.-P., Leberre R., Guery B. *et al*. (2006). Epigenetic acquisition of inducibility of type III cytotoxicity in P. aeruginosa. *BMC Bioinformatics*, vol. 7, p. 272–282.

Gallet E., Manceny M., Le Gall P., Ballarini P. (2014a, mai). Adapting LTL model checking for inferring biological parameters. In R. L. Catherine Dubois (Ed.), *Actes de la 13ème édition d'AFADL, atelier francophone sur les Approches Formelles dans l'Assistance au Développement de Logiciels, juin 2014*. AFADL.

Gallet E., Manceny M., Le Gall P., Ballarini P. (2014b, novembre). An LTL model checking approach for biological parameter inference. In S. Merz, J. Pang (Eds.), *Formal methods and software engineering*, vol. 8829, p. 155–170. Springer International Publishing.

Gastin P., Oddoux D. (2001). Fast LTL to Büchi automata translation. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01)*, vol. 2102, p. 53-65. Paris, France, Springer.

Gaston C., Le Gall P., Rapin N., Touil A. (2006). Symbolic execution techniques for test purpose definition. In *18th ifip int. conf. testcom*, vol. 3964, p. 1-18. Springer.

Kauffman S. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, vol. 22, n° 3, p. 437-467.

Khalis Z., Comet J.-P., Richard A., Bernot G. (2009). The SMBioNet method for discovering models of gene regulatory networks. *Genes, Genomes and Genomics*, vol. 3(special issue 1), p. 15–22.

King J.-C. (1975, April). A new approach to program testing. *Proceedings of the international conference on Reliable software, Los Angeles, California*, vol. 21-23, p. 228-233.

Klarner H., Streck A., Šafránek D., Kolčák J., Siebert H. (2012). Parameter identification and model ranking of Thomas networks. In *Proceedings of the 10th international conference on computational methods in systems biology*, p. 207–226. Berlin, Heidelberg, Springer-Verlag.

Mateus D., Gallois J.-P., Comet J.-P., Le Gall P. (2007). Symbolic modeling of genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*, vol. 5, n° 2B, p. 627-640.

Richard A. (2010). SMBioNet user manual, http://www.i3s.unice.fr/~richard/smbionet/ Manuel de logiciel.

Snoussi E., Thomas R. (1993). Logical identification of all steady states: the concept of feedback loop characteristic states. *Bull. Math. Biol.*, n° 55(5), p. 973-991.

Streck A. (2014). Model building for Parsybone version 2.1 Manuel de logiciel.

Thieffry D., Colet M., Thomas R. (1993). Formalisation of regulatory networks : a logical method and its automation. *Math. Modelling and Sci. Computing*, vol. 2, p. 144–151.

Thieffry D., Thomas R. (1995). Dynamical behaviour of biological regulatory networks - II. immunity control in bacteriophage lambda. *Bull. Math. Biol.*, vol. 57, n° 2, p. 277-97.

Thomas R., d'Ari R. (1990). *Biological Feedback*. CRC Press.